IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION

OF

CHRIS TUIJN

AND

PETER DE MANGELAERE

FOR

METHOD FOR SENDING MESSAGES IN A COMPUTER NETWORK

METHOD FOR SENDING MESSAGES IN A COMPUTER NETWORK

This application claims the benefit of US Provisional Application No. 60/435,254 filed December 19, 2002.

5   FIELD OF THE INVENTION

The present invention relates to a system and a method for establishing communications between a server computer and a client computer via a network such as the Internet.

10 BACKGROUND OF THE INVENTION

In the printing and publishing environment, different players interact in order to obtain print and publishing products, such as magazines, catalogues, promotional, corporate, book or specialty products in offset, flexo, screen, digital, sheet- or web-fed

15 printing. The main players that interact, in what is called in this document the "Graphic Enterprise", are the print buyer (or customer), the people in the workcenter, and the customer service representative who is the communicator between the first two main players. Different tools are used within the Graphic Enterprise,

20 such as mail servers, pre-press workflow systems (such as Apogee™ Series 3 and Apogee X from Agfa), cost estimation modules, Management Information Systems (MIS), Job Definition Format (JDF) devices, etc. However, the communication between these tools and between the different players in the Graphic Enterprise is often

25 unstructured.

There is thus a need for improved communication between the different players and the tools within the Graphic Enterprise.

SUMMARY OF THE INVENTION

30 The present invention includes a method for maintaining topology information of a topology of a system including a computer network, a first computer and a second computer connected to the first computer via the computer network, the method including (a) sending, to a bootstrap router for maintaining the topology

information and running on the first computer, a first message
indicating a modification of the topology of the system; (b)
receiving the first message by the bootstrap router; (c) updating
the topology information maintained by the bootstrap router; (d)
5   sending by the bootstrap router a second message indicating the
updated topology information to a router running on the second
computer.

Preferably, a method in accordance with the invention is
implemented by a set of computer programs. Other embodiments
10   include a method for maintaining topology information by a server
computer, and a method for maintaining topology information by a
client computer and corresponding computer programs. The invention
also includes data processing systems (such as a computer network
system, a server computer or a client computer), hardware and
15   software, for carrying out these methods, and computer readable
media having program code adapted to perform these methods.

Preferably, a project management system runs on a server
computer and application software runs on one or more client
computers, connected to the server computer via a computer network.
20   The project management system on the server computer communicates
with the software application packages on the client computers via
the network. The software applications may relate to pre-press, to
cost estimation, etc. as discussed already above. The project
management system is a communication tool that allows the different
25   players and the software applications in the Graphic Enterprise to
interact formally with each other. The different players and the
software applications may be located in different companies, e.g.
the project management software, running on the server computer, may
be in a first company while the pre-press software and the pre-press
30   team may be in another company.

In this document, a "topology" means a logical network of one
or more server computers and one or more client computers deploying
project management software (on the server computers) and client
software components (on the client computers), which software is

meant to work together. Different unrelated topologies may be deployed on e.g. the same Intranet.

In a method in accordance with the invention, it is preferred to use queues to transmit a message from one computer to another one. A queue is a sequence of messages or jobs held in auxiliary storage awaiting transmission or processing. In one embodiment of the invention, the sending computer and the receiving computer are both connected to a computer network and they both have an input queue and an output queue. Messages are transmitted between these queues of the computers. Advantageously, these queues are physically associated with the corresponding computers. That a computer has a queue that is physically associated with it means that the queue is coupled to the computer in such a way that the computer can keep utilizing the queue in case of a computer network failure, and preferably in case of any failure that is external to the computer. Queues that are physically associated with a specific computer are preferably realized by running the software, that implements the queues, on the specific computer itself, and by allocating the auxiliary storage, in which the sequence of messages or jobs of the queue is held, on the specific computer itself. In another, less preferred embodiment of the invention, the software implementing the queues does not run on the specific computer itself but on a computer (or data processing system) that is directly connected to the specific computer, i.e. not connected via the network, so that the specific computer can keep utilizing the queues in case of a network failure. An advantage of physically associated queues is that the computers can run independently of each other, e.g. a client computer can run and execute its jobs even if the server computer is down.

An advantage of a preferred embodiment in accordance with the invention is that new subsystems may be started up easily within an already established topology. It is not required to bring down the entire topology and to restart the topology after the installation of the new subsystem.

Another advantage is that diverse software applications, having their own proprietary interfaces, can be controlled by the project management software.

Further advantages and embodiments of the present invention
5   will become apparent from the following description and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

The invention is described with reference to the following drawing without the intention to limit the invention thereto, and in
10 which:

Fig. 1 diagrammatically shows an embodiment in accordance with the invention.

DETAILED DESCRIPTION OF THE INVENTION
15   Fig. 1 diagrammatically illustrates a preferred embodiment of the invention. A server computer 21 and a number of client computers 31 are interconnected by a computer network 15. Project management software runs on the server computer 21 and application software components 37, also called client software components 37 or
20 subsystems 37 elsewhere in this document, run on the client computers 31. The project management software automates the project and process management of the printing and publishing industry.

In Fig. 1, the computer network 15 is preferably the Internet or an Intranet, i.e. a portion of the Internet with restricted
25 access. The server 21 and client 31 computers may be Personal Computers (PC's) or other types of data processing systems as known in the art. In the shown embodiment, the server computer 21 has three queues that are physically associated with it: a server input queue 22, a server output queue 23 and a server controlling queue
30 24. For each hardware node 31 connected to the network 15, or client computer 31, on which one or more application software components 37 will be installed, three local queues 32 - 34 are provided that are physically associated with their client computer 31: a client input queue 32, a client output queue 33 and a client

controlling queue 34. When the project management software on the
server computer 21 requires a client software component 37 to be
run, or needs information from such a client software component 37,
a message is put on the server output queue 23. This message is
5   then got from the server output queue 23, transmitted, as indicated
by arrow A on Fig. 1, via the network 15 to the destination client
computer 31, and put on the client input queue 32 of the client
computer 31. The original message may be removed from the server
output queue 23. The message on the client input queue 32 is read
10   by the client computer 31 and an action is performed on the client
computer 31, e.g. a particular client software component 37 is run,
or information is obtained from a particular client software
component 37. Depending on the result of the action, a completion
message, such as a message indicating that the particular client
15   software component 37 ran and terminated successfully, is put on the
client output queue 33. This completion message is got from the
client output queue 33, transmitted, as indicated by arrow B on
Fig. 1, via the network 15 to the server computer 21, and put on the
server input queue 22. The completion message is read by the server
20   computer 21 and may cause an action in the server computer 21.

        As is illustrated in Fig. 1, several client software components
37 may run on the same client computer 31 (as a matter of fact,
several software components may also run on the server computer 21).
These client software components 37 often have their own proprietary
25   interface and therefore need specific software to be controlled by
the project management software on the server computer 21. This
problem may be solved as follows. For each client software
component 37 or subsystem 37 on a hardware node 31, a module that is
called in this document a subsystem plug-in 36 (see Fig. 1) is run
30   on the concerned hardware node 31. Such a subsystem plug-in 36
converts a message on the client input queue 32 to a format adapted
for the subsystem 37 that is associated to the concerned subsystem
plug-in 36 and for which the message is intended. Preferably, the
subsystem plug-in 36 scans the input queue 32 of its hardware node

31, or client computer 31, for messages that are intended for its associated subsystem 37.

Usually, the format of the messages will depend on the type of subsystem 37. Preferably, the messages are XML-based (XML stands for eXtensible Markup Language; it is a simple and flexible text format).

In Fig. 1, only a single server computer 21 is shown. More than one server computer 21 may be used. As mentioned before, in this document, a "topology" means a logical network of one or more server computers 21 and one or more client computers 31 deploying project management software (on the server computers 21) and client software components 37 (on the client computers 31), which software is meant to work together. Different unrelated topologies may be deployed on e.g. the same Intranet.

It is preferred that a specific process, called in this document a router process or simply a router, runs on each given hardware node 21, 31. This router process moves the messages in the output queue 23, 33 of the given hardware node 21, 31 to the destination input queue 32, 22. If the destination hardware node cannot be reached, the router process takes care of retries. For example the transmission of a message as indicated by arrow A in Fig. 1 is performed by a router process running on the server computer 21, while the transmission of a message as indicated by arrow B in Fig. 1 is performed by a router process running on the client computer 31.

As a special case, a router may also redirect incoming messages from the local input queue to a remote input queue; in this way, one subsystem 37 may delegate a job to another subsystem 37.

Taking care that a message is transmitted to the correct destination is preferably done by the concerned router process. This may be carried out as follows. The router process inspects all outgoing messages on the local output queue, by looking at message header properties, and subsequently moves each outgoing message to the input queue related to the correct destination. This input

queue is determined as follows. Preferably the software on the main
server 21 and the subsystems 37 on the client servers 31 communicate
with each other through each other's unique system names. The
router then translates such a system name, that indicates the
destination, e.g. a specific subsystem 37, to the corresponding
input queue. This implies the need for some mechanism to get all
routers informed at all times of the locations of all subsystems 37,
and of the software running on the server computer 21. This
mechanism includes a special router and the controlling queues 24,
34 that were mentioned already above in the embodiment shown in
Fig. 1. The special router, called the bootstrap router, provides
all the other routers with the information concerning the locations.
The bootstrap router has the same functions as the other routers,
which were already discussed above, and additionally the bootstrap
router maintains the topology information of the system, i.e. where
queues are provided, where routers are running, which (sub)systems
run on what hardware nodes 21, 31, etc. When a new subsystem 37 is
first started, a message is put on the controlling queue 34 of the
hardware node 31 of this subsystem 37. The router of this hardware
node 31 transmits this message to the controlling queue of the
bootstrap router. The bootstrap router then updates its topology
information and makes this information available to all routers in
the system. The bootstrap router may run on a server computer 21 or
on any other computer.

An advantage of this "bootstrapping mechanism" is that new
subsystems 37 may be started up easily within an already established
topology. It is not required to bring down the entire topology and
to restart the topology after the installation of the new subsystem
37. Moreover, also a new client computer 31 with corresponding
router and queues 32, 33, 34 can dynamically become part of an
already established topology.

Another advantage is that subsystems 37 can easily "travel" to
other client computers 31. It is not necessary to bring down the
entire topology and to restart the topology after having moved the

subsystem 37. If e.g. a given server 21 or client 31 computer
fails, the software on that computer may have to be relocated to one
or more other computers. The bootstrapping mechanism then takes
care of informing all involved parties (e.g. all other subsystems
5 37) of the new locations.

Preferably only a single input queue 22, 32 is used per
computer 21, 31 for all software applications that are part of the
concerned topology. Thus, all client software components 37 on a
specific client computer 31 make use of a single, shared input queue
10 32 (of course, 'all client software components' only means the
client software components 37 that are part of the topology, and
that thus communicate with a server computer 21 via messages; it
does not include some arbitrary software that may also run on the
specific client computer 31). More preferably, also a single shared
15 output queue 23, 33 is used. Most preferably, each server computer
21 and each client computer 31 has a single set of queues associated
with that computer and to be used for all software applications that
are part of the concerned topology and that run on that computer.
The set of queues includes a single shared input queue 22, 32, a
20 single shared output queue 23, 33 and a single shared controlling
queue 24, 34. An advantage of such a queue architecture is that it
alleviates administration. Suppose that each subsystem 37 would
read from its own queue instead of reading from a shared queue
associated with the client computer 31 that the subsystem 37 is
25 running on. In that case, adding a new subsystem 37 would require
introducing the deployment of new, previously non-existing queues
into an already up-and-running configuration.

Preferably, the queues are persistent queues; this means that
their contents is preserved, even in case the computer system goes
30 down for some reason. The subsystems 37 preferably log their
results locally. Results that are needed by other subsystems 37 or
by project management software on the server computer 21 are
preferably put on the corresponding client output queue 33. An

advantage is that the subsystems 37 can run independently of each other, even if the server computer 21 is down.

5   Those skilled in the art will appreciate that numerous modifications and variations may be made to the embodiments disclosed above without departing from the scope of the present invention.

List of reference signs

10 : computer network system

15 : network

5   21 : server computer

22 : server input queue

23 : server output queue

24 : server controlling queue

31 : client computer

10   32 : client input queue

33 : client output queue

34 : client controlling queue

36 : subsystem plug-in

37 : client software component

15   A : arrow

B : arrow